End-to-End Simulation of On-Orbit-Servicing: Technical implementation of communications

Daniel Weber* Marcin Gnat† Armin Hauke‡

Felix Huber§

DLR, Oberpfaffenhofen, Bavaria, 82234, Germany

Carlos G. Acero¶

LSE Space GmbH, Weßling, Bavaria, 82234, Germany

The robotic on-orbit servicing technology promises an increase in life-time of operational satellites and the removal of space debris. Though such tasks are of high importance for future space exploration in general, many agencies and companies struggle with running such missions. One of the reasons is definitely to find the ultimate business case. Moreover, on-orbit servicing imposes very high risks on a mission due to its complexity which is almost as high as in human spaceflight. It is therefore essential to perform end-to-end hardware-in-the-loop simulations of a mission on ground before it is being carried out in space. For this purpose, the On-Orbit-Servicing End-to-End Simulation project (OOS E2E) has been established at the German Aerospace Center (DLR). The project uses expertise, resources and facilities from a couple of DLR institutes. Contributors are the German Space Operations Center (GSOC), the Institute of Robotics and Mechatronics (RM) and the Institute of System Dynamics and Control (SR). The aim of the On-Orbit-Servicing End-to-End Simulation project is to connect the different simulation facilities of these institutes and integrate them into a single end-to-end simulation of on-orbit servicing. One of the facilities is the European Proximity Operations Simulator (EPOS) to simulate the rendezvous maneuver between the client satellite and the chaser satellite. The other facility is the On-Orbit-Servicing Simulator (OOS-Sim) of the Institute of Robotics and Mechatronics to simulate the robotic telepresence operations.

In this paper we focus on the implementation of data communication between all of the simulation facilities. Especially, the need for real-time robotic telepresence operations creates a new set of requirements for the communication chain. To account for a real-world scenario, it is therefore important to simulate the communication chain and the operational environment of an on-orbit servicing mission. The behavior of the space link, as well as the data transportation on ground, must be taken into account, including all communication parameters like possible loss, delay, jitter, corruption or duplication in the TM/TC data streams. As these parameters vary over time, the occurrence of bursts and the timely distribution of these parameters play a significant role. Furthermore, the beginning, the end and possible handovers of a satellite passage must be simulated. As the robotic telepresence operations are as important as the housekeeping operations, the setup must be optimized for processing both robotic real-time data and standard satellite TM/TC data in parallel. To do so, both data streams must be multiplexed into a single space link. This is done by specially developed FPGA devices that can be synchronized to a common master clock to multiplex/demultiplex both data streams into/from a single space link in a timely manner. Furthermore, Space Link Protocols have to be implemented between the space and ground components of the simulation. In the same way, the protocols of the ground segment must be optimized for the processing of real-time data.

*Software Engineer, Communication and Ground Stations, DLR Oberpfaffenhofen, Germany, Daniel.Weber@dlr.de.
†Ground Data Sys. Manager, Communication and Ground Stations, DLR Oberpfaffenhofen, Germany, Marcin.Gnat@dlr.de.
‡Software Group Manager, Communication and Ground Stations, DLR Oberpfaffenhofen, Germany, Armin.Hauke@dlr.de.
§Institute Director, Space Operations and Astronaut Training, DLR Oberpfaffenhofen, Germany, Felix.Huber@dlr.de.
¶Systems Engineer, Communication and Ground Stations, DLR Oberpfaffenhofen, Germany, Carlos.GarciaAcero@dlr.de.

For this purpose, a common Space Link TM/TC library has been developed in C++, which is being shared between the European Proximity Operations Simulator (EPOS), the On-Orbit-Servicing Simulator (OOS-Sim), the satellite simulator, the dynamic simulator, the robotic console, the rendezvous console, the satellite console and the standard TM/TC chain for housekeeping. To operate the distributed simulation system in a reliable way, it is further necessary to implement a monitoring and control software. For this purpose, we use an instance of an already established Antenna Monitoring and Control Framework that is used at the Ground Station Weilheim.

In this paper we present the technical implementation of the communication chain of the project and results of performance test measurements. In particular we analyze the real-time requirements for the setup. Finally, we discuss future improvements and how the setup can be adopted to a real-world scenario. A demonstration video of a full end to end simulation is also presented.[1]

I. Introduction

On-orbit servicing is essential for space missions: The Hubble Space Telescope has been launched in 1990 and has been serviced in five Space Shuttle missions in order to fix significant technical issues.[2] Solar Arrays were replaced, corrective optics and coprocessors for the flight computer were installed.[2] In order to build the International Space Station it took 134 spacewalks and 900 hours of EVA activities.[2]

First steps towards a replacement of EVA activities by (tele-) robotic operations were taken early on with ROTEX in April 1993. This was the first robot in Space and it was a part of the Spacelab D2 Mission of the German Aerospace Center (DLR).[3–9] In 1997 the National Space Development Agency in Japan (NASDA) launched the Engineering Test Satellite VII (ETS-VII). Robotic on-orbit servicing between a chaser and a target spacecraft was demonstrated.[10–16] The chaser vehicle used a 2 m long robotic arm to grab the client. Rendezvous and Docking were demonstrated. Also teleoperation was performed to install Orbit Replacement Units (ORUs).[11] Another robotic on-orbit servicing was demonstrated by the Orbital Express mission by the Defense Advanced Research Projects Agency (DARPA).[17–19] The payload was launched in 2007 and consisted of two spacecraft. It was possible to demonstrate autonomous fuel transfer as well as the autonomous installation of ORUs, such as the replacement of batteries and the flight computer.

Astronauts at the ISS have performed tasks like releasing solar array panels that got stuck during deployment or removing and replacing failed components. Today, the robotic Space Station Remote Manipulator System (SSRMS) can perform servicing tasks.[2] The Canadian Special Purpose Dexterous Manipulator (SPDM) or Dexter is used to do precise telerobotic operations.[2,20] The remote manipulator system (Canadarm2) can be used to grasp and berth visiting vehicles. Also telerobotic operations from earth have been carried out at the ISS.[21] An example of a current experiment is the Robotic Refueling Mission 3. It shall perform cryogenic liquid methane transfer as well as xenon gas transfer and will be launched in 2018.[22]

While the feasibility of robotic on-orbit servicing has been demonstrated in science for many times, it is not being used in commercial space business as there is a high initial risk and cost in development of this technology. An intuitive thought might be to avoid launching a robotic service satellite and instead launch one of the operational satellites. However, this argumentation is highly misleading, as the financial benefit depends on the number of satellites to be serviced and the extension of life-time that can be achieved per serviced satellite.[2] As private companies like SpaceX have introduced new business models with low costs and innovative technologies like the reuse of booster rockets, past financial calculations must be reconsidered.[23] Apart from commercial aspects, it is also essential to remove existing space debris, otherwise the cascading collisions of debris particles will render space activities infeasible in specific orbital ranges.[24,25]

In order to advance robotic on-orbit servicing technologies – especially in the commercial sector – it is necessary to provide the means to test and verify procedures on ground before they are executed in space. The End-to-End On-Orbit-Servicing Simulation project copes with that by providing means for a HIL supported simulation of the complete servicing procedure. The components of the simulation environment are: The European Proximity Operations Simulator (EPOS) which simulates the rendezvous maneuver. The OOS-Sim to simulate the robotic servicing.[26] The Satellite Simulator which simulates the communication infrastructure of the satellite and numerically calculates the flight dynamics. Furthermore the space link and ground station must be simulated. Also a control room environment must be provided for the operators of the simulated scenario.

The telerobotic operations are considered one of the core aspect of on-orbit servicing missions. Thus

the communication chain becomes an integral part of the whole simulation. The communication system is simulated to provide realistic communication conditions. This includes the control room, the ground station and the space link as well as the protocols spoken in between. Depending on the simulation scenario the delay can range from one hundred milliseconds in case of a low Earth Orbit (LEO) to several seconds in case of using a geostationary relay satellite. Also Loss of Signal (LOS) and Acquisition of Signal (AOS) must be simulated. For this purpose a WAN-Simulator is used.[27]

To achieve optimal soft real-time conditions, previous telerobotic missions bypassed the existing communication infrastructure by using a dedicated space link. Otherwise they would have suffered from a higher delay and jitter of the existing communication infrastructure which is typically not optimized for soft real-time operations but rather for guaranteed delivery. The existing protocols in the communication chain must therefore be analyzed and optimized. For this purpose a setup has been proposed that multiplexes the commands of the standard satellite console with the robotic soft real-time telecommands with the help of a real-time FPGA device.

II. Overview of the communication chain

The aim of the communication setup is to implement the communication to the satellite in a single simulated space link supporting soft real-time delivery while maintaining the possibility to carry out standard satellite operations with guaranteed delivery in parallel. For this purpose the telecommands from the satellite console (SACO) are multiplexed with the telecommands of the robotic console (ROCO) by a FPGA based merging device (MEGI) using the UDP/IP protocol.

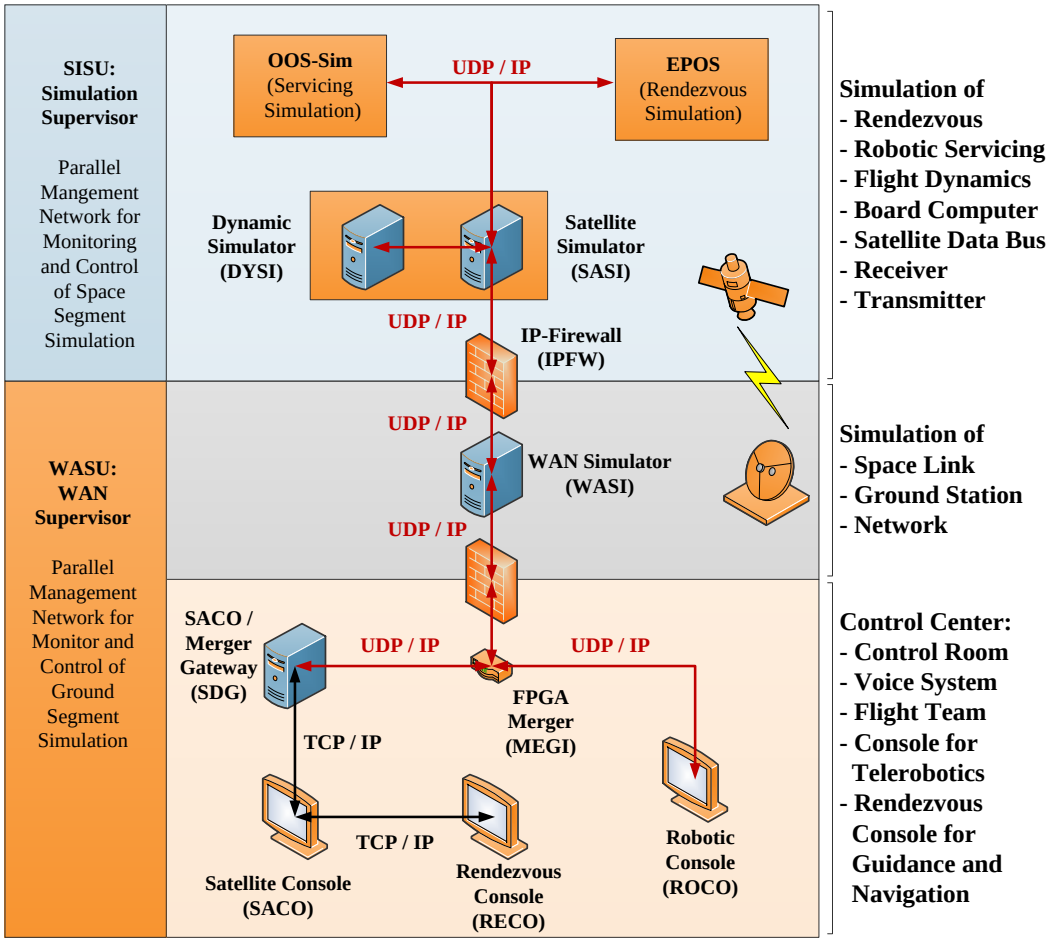

Figure 1. Overview of the simulation setup. A FPGA merger (MEGI) multiplexes telecommands at a total frequency of 400 Hz. ROCO uses a frequency of 200 Hz while RECO and SACO share the other 200 Hz. The SDG acts as a gateway to convert the TCP/IP packets of SACO to UDP/IP. WASI simulates communication parameters of the communication infrastructure. SASI simulates the satellite communication infrastructure and contains a dynamic simulator (DYSI) which calculates the flight dynamics of the system. The OOS-Sim and EPOS perform HIL simulations.

II.A. Overview of the communication chain in the control center setup

The satellite console (SACO) is a Linux workstation that is used in multi-mission satellite operations. To communicate with a spacecraft, it uses the Spacecraft Operating System (SCOS) which itself runs on a virtual machine on a VMware ESXi Server. The SCOS uses TCP/IP connections to exchange telecommands and telemetry with a ground station. As the setup must meet the soft real-time requirements for telerobotic operations, the TCP/IP packets are converted to UDP/IP by a gateway (SDG). Due to the fact that there is no real ground station within the simulation setup, the gateway must also simulate acknowledges for SCOS providing information on telecommand radiation by the ground station.

Figure 2. Control room with the telepresence master device based on the DLR Light-Weight Robot (LWR). The control room is virtualized. The left activated terminal is used for voice communication.

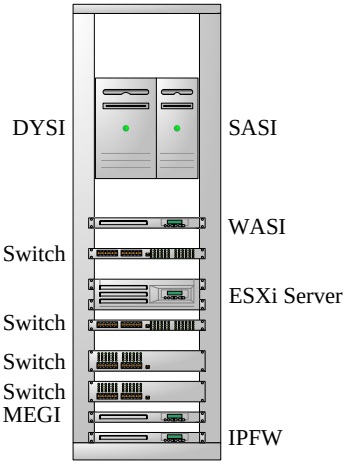

Figure 3. Overview of the server rack that implements the communication setup. The virtualized consoles run on a VMware ESXi server.

The robotic console (ROCO) consists of the telepresence master device which is based on the DLR Light-Weight Robot (LWR) as well as a software component which creates telecommands and can read telemetry. For this purpose the CCSDS Space Communication Protocols are implemented in a TM/TC library written in C++11, which is the programming language of choice for all the simulation facilities.

The rendezvous console (RECO) is installed on a virtual machine on a VMware ESXi server. To read telemetry and to create telecommands it uses the same TM/TC C++11 library. The console interacts with the on-board guidance, navigation, and control (GNC) system of the satellite. In order to receive the telemetry and to send telecommands it shares its bandwidth with SACO and uses an external interface (EXIF) of SCOS.

The FPGA merger (MEGI) is a custom developed device based on a Xilinx Spartan 6 FPGA. It alternately takes UDP/IP packets from SDG and ROCO and multiplexes them into the uplink. The bandwidth is limited by the reference scenario. To keep in line with these bandwidth restrictions, the telecommands (CLTUs) may have a maximum length of 74 octets while the telemetry has a fixed length of 1115 octets. The telecommands and telemetry is forwarded at intervals of 2.5 ms.

Max. Delay	Max. Jitter	Max. Uplink Bandwidth	Max. Downlink Bandwidth
100 ms	2.5 ms	256 kbit/s	6 Mbit/s

Table 1. Communication requirements of the simulated reference scenario

II.B. Overview of the communication in the space link and ground station simulation

The space link and ground station infrastructure is simulated by creating jitter, delay, loss, corruption, duplication and reordering of the UDP/IP packets. For this purpose a WAN-Simulator is implemented. The WAN-Simulator uses the Traffic Control System of the Linux kernel in combination with a special queueing discipline (qdisc) called netem.[28] As only internal kernel IRQs can interrupt this system, the timing of the system is very precise. To optimize the temporal resolution of the WAN-Simulator, the high resolution

timers must be enabled in the Linux kernel. Also the internal clock frequency of the kernel should be set to CLOCK_HZ = 1000 Hz. Bursts can be implemented by time correlation. To simulate Acquisition of Signal (AOS) or Loss of Signal (LOS), the loss can be set to zero or 100 percent. The timing of the WAN Simulator is precise to the sub-millisecond regime. Detailed measurements can be found in a previous study.[27]

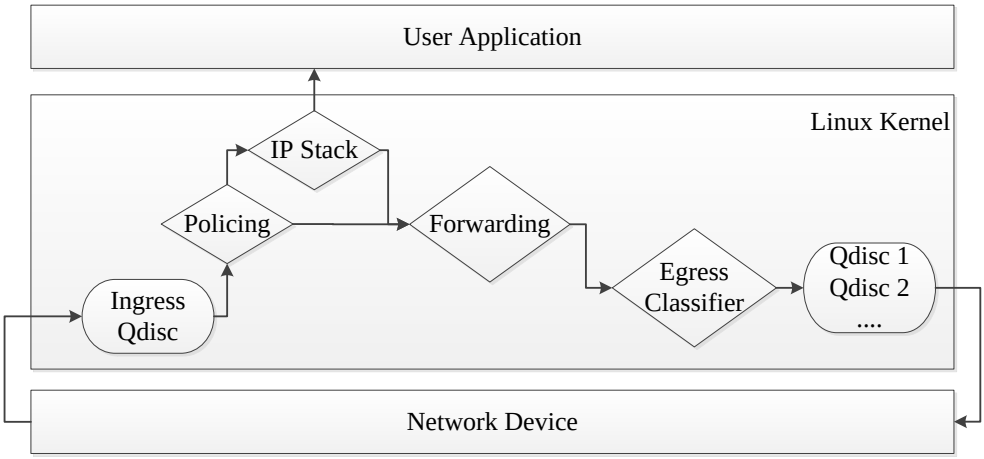

Figure 4. The Linux Traffic Control System can filter ingress traffic (Policing) as well as egress traffic. So called qdiscs (Queueing Disciplines) can be installed to schedule the traffic. A special egress qdisc is netem, which implements various communication parameters like delay and jitter.[28]

II.C. Overview of the communication chain in the space segment of the simulation

The space segment consists of three simulation facilities, the OOS-Sim and EPOS as well as SASI. EPOS simulates the rendezvous maneuver between the client and the chaser, while the OOS-Sim simulates the robotic interaction between the two satellites. SASI consists of two physical workstations. The first is a VxWorks workstation that emulates the satellite communication infrastructure and subsystems. The second is a Microsoft Windows workstation which performs numerical calculations in real-time in order to determine the orbit dynamics and the relative position between the chaser and the client satellite. The forces acting between the two satellites during the operations are sent by EPOS and OOS-Sim to SASI at a frequency of 200 Hz. As this data is needed for the numerical calculations the data is sent in fixed time intervals. The UDP/IP protocol is used to meet the soft real-time requirements of the setup.

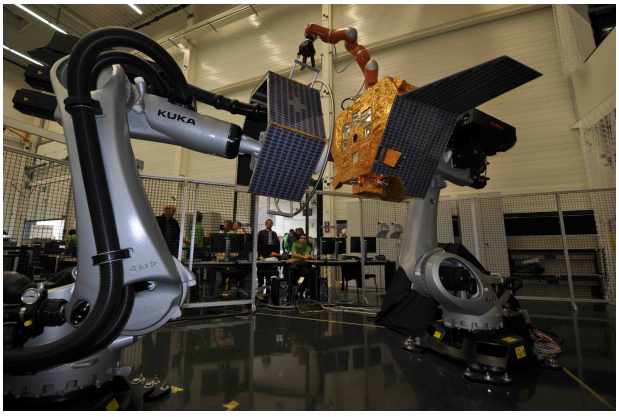

Figure 5. OOS-Sim. A robotic arm is mounted on the chaser satellite to service the client satellite.

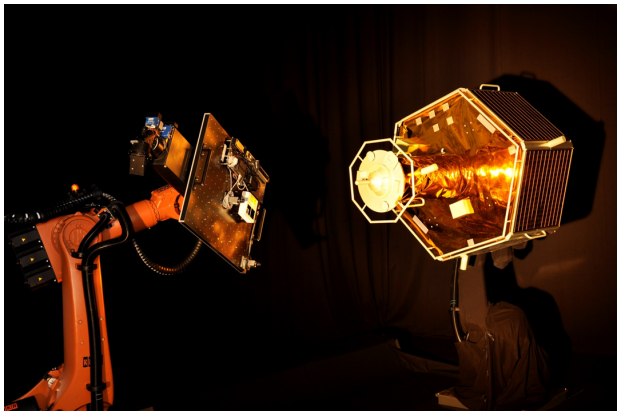

Figure 6. EPOS simulates the rendezvous between the chaser and the client satellite.

III. Optimization of the communication chain with real-time FPGA devices

III.A. The FPGA Merger (MEGI)

The merger (MEGI) comprises a Xilinx Spartan 6 FPGA, four PHYs and one separated embedded controller for commanding and monitoring. The algorithms were implemented in the Handel-C language that allows for high level programming and simulation of the functionality while still providing access to the low level primitives within the FPGA. The PHYs provide the physical connection and deliver their data to four soft MAC cores that are realized within the FPGA. The two receiving MAC cores run in their own clock domain derived from the PHYs, while the main function and sending MACs run in a clock domain locked onto an external 10 MHz reference. In order to achieve the strict 5 ms timing, the main function generates a 400 Hz trigger that alternately activates the reading of the two incoming packet streams.

Since the receiving MACs run in their clock domain, synchronizing FIFOs transfer the packet data between the clock domains. The FIFO control logic is transparently created by the compiler. At each trigger, the corresponding FIFO is checked for a data packet. If none is there, either an idle packet is sent in the case of robotic commands or no packet is sent. Due to the generation of the trigger pulses and packet assembly in hardware, strictly no jitter in the timing is generated. The transmit packets are sent through one of the redundant output ports.

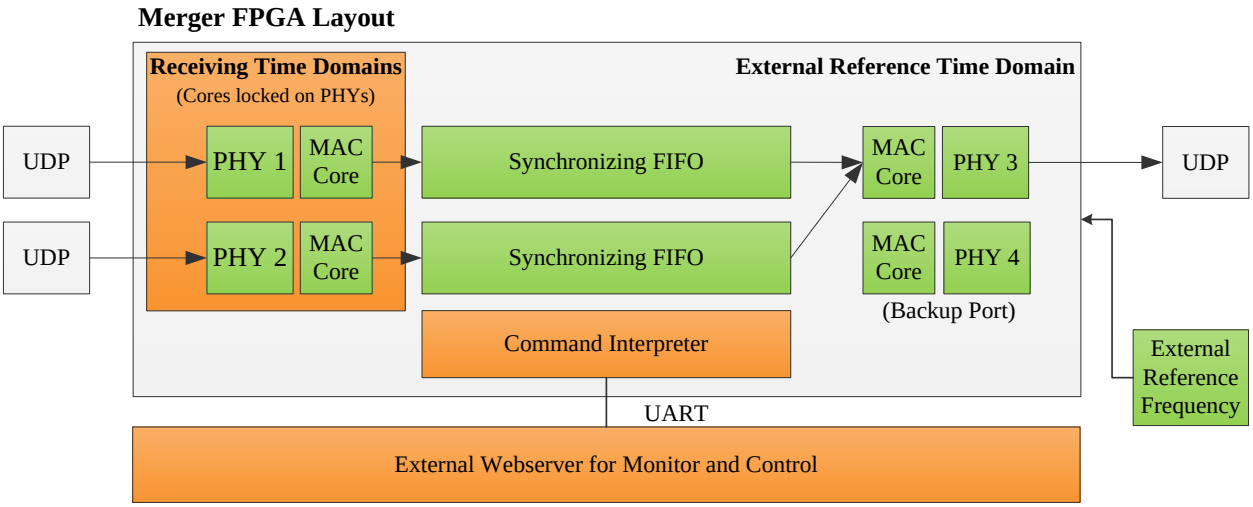

Figure 7. Logical layout of the FPGA merger (MEGI). UDP packets are received by two PHYs and are forwarded to a MAC Core. The cores are locked to the frequency of the PHYs. The data of the packets is forwarded via synchronizing FIFOs to the sending time domain. A MAC Core alternately takes packages from both FIFOs and forwards them every 2.5 ms.

All the parameters and telemetry of the system can be remotely controlled with the independent embedded controller. For security reasons, the M&C Ethernet connection is separated from the FPGA and the commands are forwarded to the FPGA via a UART interface. The main function in the FPGA has a command interpreter for sending commands and reading out telemetry such as IP and MAC addresses, port numbers, packet counters etc. Due to the parallelism within the FPGA, these functions cannot affect the timing of the packet generation.

III.B. The IP-Firewall (IPFW)

Another FPGA is used to implement the IP Firewall (IPFW). As it is based on the same architecture it can be locked to the same reference frequency as the FPGA merger (MEGI). This can be used to improve the real-time capability of the communication system and to reduce the jitter to a very low amount. Since all synthesized function blocks within the FPGA can run in parallel, the incoming packets are already inspected while being received. Dedicated processor cores have been synthesized to implement basic IP firewall rules. The FPGA also allows for a firewall functionality like protecting against denial of service and buffer overflows as in a hardware-only design no such failures can occur. The main functions also handles the IP protocol requirements like ARP, and MAC address tables.

IV. Remote monitoring and control system for the communication chain

Figure 8. Example layout of the GUI front-end. Several monitors are used to show the status of the devices in the communication system of the simulation.

The software used to monitor and control the setup uses an architecture with a central server as well as several generator processes. The generators connect to the server and to the devices and generate monitoring information for the server. The server stores the monitoring information and forwards it to the WAN supervisor (WASU). The WAN supervisor can command the devices by setting parameters. The software framework is based on a software to control the antennas at the ground station in Weilheim.[29] Three generators have been developed for SDG, MEGI and WASI. A workflow engine of the framework can be used to send commands at specific points in time without the need of interference of the operator. This enables to change the communication parameters at the WAN-Simulator (WASI) for AOS and LOS.

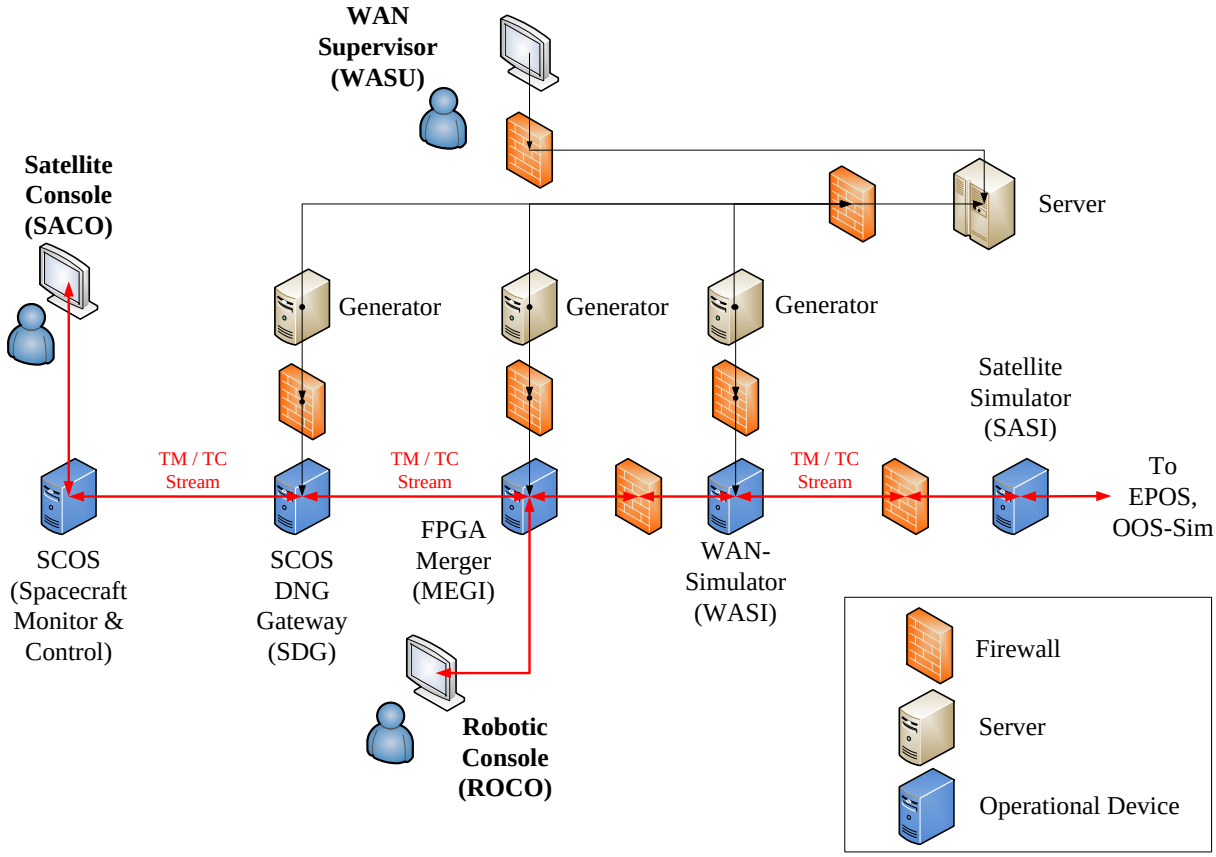

Figure 9. Software architecture of the remote monitoring and control system. A server holds the monitoring and control parameters. Generators create the parameters by connecting to a device and sending them to the server. Generators can send commands to the devices. A GUI is used to visualize the parameters and to trigger commands.

V. Implementation of CCSDS Space Communication Protocols in the communication chain

Synchronization and Channel Coding Layer			
TM		TC	
Frame Synchronization	✓	BCH(63,56)-Coding	✓
Randomization	✓	Randomization	✓
RS(255,223)-Coding	✓	CLTU Synchronization	✓
Turbo Coding	∅	LDPC Coding	∅
LDPC Coding	∅		
Convolutional Coding	∅		
Data Link Layer			
TM		TC	
Master Channel	✓	Master Channel	✓
Virtual Channels	✓	Virtual Channels	✓
Secondary Header Access	✓	CRC Calculation	✓
OCF Access	✓	COP (at Satellite)	✓
		MAP Service	∅
Space Packet Protocol			
Packet Service		✓	
Octet Service		✓	

Table 2. Implemented features in the C++11 TM/TC library

The satellite communication protocols used in the simulation, range from the lower physical layer to the higher application level. These are defined by CCSDS standards: The Space Packet Protocol, the TM/TC Data Link Layer and the TM/TC Synchronization and Channel Coding Layer.[30–34] To simulate the whole communication chain it is necessary that all simulation facilities implement the protocols correctly. Therefore a library is written in C++11 which is shared between all simulation facilities. This ensures the re-usability of the setup with a real telerobotic mission. The features of the lowest physical layer (like PLOP) are not simulated as they can only be implemented by sophisticated radio frequency equipment.

V.A. Space Packet Protocol

The Space Packet Protocol is used for routing service data units (SDUs) between different applications on the satellite and on ground. Space Packets are variable in length and don't have any timing restrictions. The Space Packets can be segmented. However, the communication infrastructure benefits from small unsegmented Space Packets because of the soft real-time requirements of the setup. Lost segments transferred via UDP/IP can cause a retransmission overhead.

V.B. TM Data Link Protocol

The Data Field of the TM Transfer Frame contains Space Packets. The OCF is used to store a CLCW. It is used to determine if it is necessary to retransmit telecommands. The soft real-time haptic-feedback data of the robotic arm on the satellite is transferred within the DATA field of the Secondary Header. The provision of the Secondary Header is set for all virtual channels. This enables transmission of the soft real-time haptic-feedback data in fixed time intervals.

V.C. TM Synchronization and Channel Coding

For transmission of the TM Transfer Frames, the TM Transfer Frame may be encoded with Reed-Solomon ($E = 16$, $I = 5$, $J = 8$).[33] Therefore the size of a TM Transfer Frame is fixed to 1115 octets. In the simulation, the Reed-Solomon encoding can be enabled or disabled. If Reed-Solomon encoding is enabled,

$2EIJ$ bits of check symbols are appended to the TM Transfer Frame.[33] A randomizer/derandomizer can be applied to the TM Transfer Frames to increase the bit shift density.[33] Also an Attached Sync marker can be put in front of the TM Transfer Frame for Frame Synchronization.[33] In the current configuration TM Transfer Frames of 1115 octets are created at the satellite simulator (SASI) and one TM Transfer Frame is forwarded by one UDP/IP packet. Frame Synchronization is done by reading the beginning of a UDP/IP packet and associating the content with exactly one TM Transfer Frame.

V.D. TC Data Link Protocol

For telecommanding several virtual channels can be selected. One of the channels is reserved for robotic soft real-time telecommands. In this channel, the bypass flag is set, to bypass the Control Operations Procedure (COP). The standard satellite commands can use the COP to ensure the execution and retransmission of telecommands. The ECF is being used for error correction. Furthermore the TC Transfer Frames have a maximum size of 56 octets. The TC Transfer Frames can be segmented, but it is recommended to use as little segmentation as possible. Lost segments can cause a retransmission overhead when used in combination with UDP/IP.

V.E. TC Synchronization and Channel Coding

The CLTU has a Start Sequence and a Trailer for Bit Synchronization. Inside the Data Field the TC Transfer Frame is encoded into BCH code blocks. In order to meet the soft real-time requirements of the telerobotic operations it is necessary to send commands every 5 ms. To give the standard satellite operations an equal priority it is therefore necessary to send one CLTU every 2.5 ms. This is done alternately from SACO and ROCO. At a uplink rate of 250 kbit/s, a maximum CLTU size of 74 octets is chosen, which results in approximately 240 kbit/s.

VI. Integration and Testing

The setup has been fully integrated and tested. The communication chain is regularly being used to transfer the telemetry and telecommands of the consoles. Science data like the camera images of the GNC system and the robotic telepresence data is regularly being transferred with the setup. The development of the setup is advanced in weekly SCRUM sprints and the simulation of the reference scenario is executed at the end of each weekly sprint.

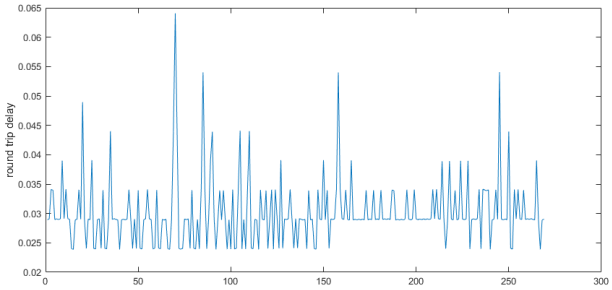

Figure 10. Round trip time of telerobotic space packets in seconds. The WAN-Simulator is set to a delay of 0 ms. The measurements show a delay of about 30 ms. Only every tenth space packet is measured.

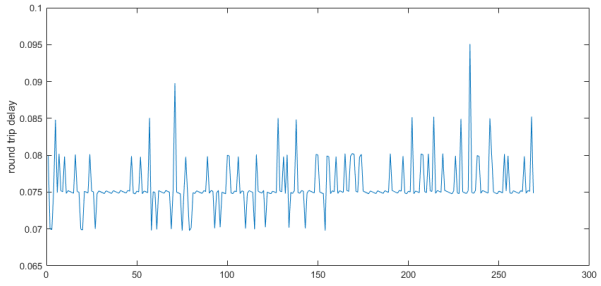

Figure 11. Round trip time of telerobotic space packets in seconds. The WAN-Simulator is set to a delay of 50 ms. The delay adds up to a total amount of about 75 ms. Only every tenth space packet is measured.

While all the components of the communication chain have been tested in detail on its own, only some first measurements were carried out of the complete system. These measurements show that the WAN-Simulator operates as expected. The plots show the round trip time of the telerobotic operations from the telerobotic master device on the ground to the robot in space. The round trip time is calculated for every tenth packet. As can be seen, the measurement results are discrete with step sizes of around 5 ms. A possible reason for this, is that the ROCO software is currently running on a virtual machine. This must be changed in the future. However these measurements were taken only as a validation of the implementation and as a proof of concept. The timing optimization of the setup has just started. A more detailed timing analysis

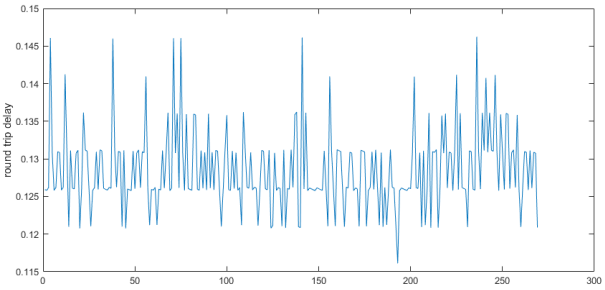

Figure 12. Round trip time of telerobotic space packets in seconds. This measurement was carried out with a delay of 100 ms at the WAN-Simulator. Also an additional jitter was imposed by the WAN-Simulator, but this cannot be analyzed in detail because only every tenth packet was measured.

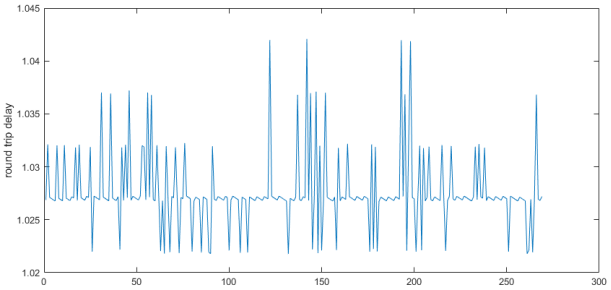

Figure 13. Round trip time of telerobotic space packets in seconds. In this measurement the WAN-Simulator was set to a delay of 1000 ms. Only every tenth space packet is measured.

must be carried out in the future.

In particular it will be necessary to analyze how well the setup and the protocols can cope with lost UDP/IP packets in the communication chain. This is especially important for the standard satellite operations who usually rely on guaranteed delivery. In this setup only the COP mechanism of the CCSDS Data Link Layer ensures the delivery of TC Transfer Frames to the satellite. Therefore it must be analyzed up to which loss conditions it is still feasible for operators on ground to only rely on the COP mechanism for the guaranteed delivery of telecommands. Depending on the needs of the mission, the consequences could then either be to reduce the package loss by improving the communication infrastructure or to use an additional SLE link and create the real-time data stream at the ground station.[35]

VII. Conclusion

A simulation setup for a HIL-OOS Simulation has been implemented and the setup meets the delay requirements of the simulation scenario. The jitter requirements still need to be validated. It is still necessary to lock the FPGA devices to an external reference frequency and to analyze the jitter in detail. It has been shown that the communication parameters can be changed for other scenarios using the WAN-Simulator. The necessary CCSDS protocols have been implemented in a C++11 library and thus enable a complete simulation of the communication to the spacecraft. The whole setup can be monitored and controlled with a software framework by one operator.

The setup also shows that it is possible to lock several electronic devices in an Ethernet communication chain to a fixed reference frequency. By implementing these devices on a hardware level, it is possible to precisely control the timing in the communication path, though this is not an inherent feature of Ethernet in principle. An example is the implementation of a standard component like the IP Firewall (IPFW) with a FPGA device. By repeating this procedure with other components, the whole data flow is manageable and synchronized to an external frequency. Advancing this concept even further, the reference frequency could be modulated on the uplink and used by the spacecraft as well. This setup therefore combines the advantages of a distributed non real-time Ethernet network between all ground stations with a fixed external reference frequency synchronizing and controlling the data flow in detail. Using a GNSS signal as a reference frequency is the best choice as it is easily available and very stable. While in the past, it was possible to use communication infrastructure providing guarantees on delay and jitter (Token Ring, ATM), today most of the communication infrastructure is replaced with Ethernet.

VIII. Appendix: Acronym List

APID	Application Process Identifier
ATM	Asynchronous Transfer Mode
BCH	Bose-Chaudhuri-Hocquenghem
CCSDS	Extravehicular Activity
CLCW	Communications Link Control Word

CLTU	Communications Link Transmission Unit
COP	Control Operation Procedure
DARPA	Defense Advanced Research Projects Agency
DEOS	Deutsche Orbitale Servicing Mission
DNG	Deterministic Network Gateway (former name for the merger MEGI)
E2E	End to end
ECF	Error Control Field
EPOS	European Proximity Operations Simulator
ETS-VII	Engineering Test Satellite VII
EVA	Extravehicular Activity
FARM	Frame Acceptance and Reporting Mechanism
FIFO	First-In First-Out
FOP	Frame Operation Procedure
FPGA	Field Programmable Gate Array
GNC	Guidance Navigation and Control
GNSS	Global Navigation Satellite System
GUI	Graphical User Interface
HIL	Hardware in the Loop
IP	Internet Protocol
IPFW	IP Firewall
IRQ	Interrupt Request
ISS	International Space Station
LDPC	Low-Density-Parity-Check-Code
LWR	Light-Weight Robot
MC	Master Channel
MEGI	Merger/Distributor
NASDA	National Space Development Agency of Japan
NCTRS	Network Control and Telemetry Routing System
OCF	Operational Control Field
OOS	On-orbit servicing
OOS-Sim	On-Orbit-Servicing Simulator
ORU	Orbital Replacement Unit
PCS	Payload Control System
PHY	Physical Layer (Integrated Circuit)
PLOP	Physical Layer Operations Procedure
SASI	Satellite Simulator
SCID	Spacecraft ID
SCOS	Spacecraft Operating System
SDG	SCOS DNG Gateway (DNG was the former name of the merger MEGI)
SDU	Service Data Unit
SLE	Space Link Extension
SSRMS	Space Station Remote Manipulator System
TCP	Transport Control Protocol
UART	Universal Asynchronous Receiver Transmitter
UDP	Unidirectional Datagram Protocol
VC	Virtual Channel
VCID	Virtual Channel ID
VSN	Version Number
WAN	Wide Area Network
WASI	WAN-Simulator

References

[1]DLR Youtube Channel, "On-Orbit Servicing - End-to-End Simulation," https://www.youtube.com/watch?v=ADS-qGI5kOc, 2018.

[2]NASA Goddard Space Flight Center, "On-Orbit Satellite Servicing Study," 2010.

[3]Hirzinger, G., "The space and telerobotic concepts of the DFVLR ROTEX," *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, Vol. 4, IEEE, 1987, pp. 443–449.

[4]Hirzinger, G., Dietrich, J., Gombert, B., Heindl, J., Landzettel, K., and Schott, J., "The sensory and telerobotic aspects of the space robot technology experiment ROTEX," *Proc. i-SAIRAS'92*, 1992.

[5]Hirzinger, G., Brunner, B., Dietrich, J., and Heindl, J., "Sensor-based space robotics - ROTEX and its telerobotic features," *Robotics and Automation, IEEE Transactions on*, Vol. 9, No. 5, 1993, pp. 649–663.

[6]Hirzinger, G., "ROTEX - The first space robot technology experiment," *Experimental Robotics III*, Springer, 1994, pp. 579–598.

[7]Hirzinger, G., Landzettel, K., Heindl, J., and Dietrich, J., "ROTEX - the first robot in space," Tech. rep., SAE Technical Paper, 1994.

[8]Hirzinger, G., Brunner, B., Dietrich, J., and Heindl, J., "ROTEX - the first remotely controlled robot in space," *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, May 1994, pp. 2604–2611 vol.3.

[9]Brunner, B., Hirzinger, G., Landzettel, K., and Heindl, J., "Multisensory shared autonomy and tele-sensor-programming-key issues in the space robot technology experiment ROTEX," *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*, Vol. 3, IEEE, 1993, pp. 2123–2139.

[10]Yoshida, K., "ETS-VII Flight Experiments For Space Robot Dynamics and Control," *The International Journal of Robotics Research*, Vol. 22, No. 2, 2003, pp. 321–335.

[11]Imaida, T., Yokokohji, Y., Doi, T., Oda, M., and Yoshikawa, T., "Ground-space bilateral teleoperation of ETS-VII robot arm by direct bilateral coupling under 7-s time delay condition," *IEEE Transactions on Robotics and Automation*, Vol. 20, No. 3, June 2004, pp. 499–511.

[12]Landzettel, K., Brunner, B., Deutrich, K., Hirzinger, G., Schreiber, G., and Steinmetz, B., "DLR's experiments on the ETS VII space robot mission," *Proceedings of the 9th ICAR*, 1999.

[13]Oda, M., "Experiences and lessons learned from the ETS-VII robot satellite," *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, Vol. 1, 2000, pp. 914–919 vol.1.

[14]Yoshida, K., Hashizume, K., and Abiko, S., "Zero reaction maneuver: Flight validation with ETS-VII space robot and extension to kinematically redundant arm," *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, Vol. 1, IEEE, 2001, pp. 441–446.

[15]Inaba, N. and Oda, M., "Autonomous satellite capture by a space robot: world first on-orbit experiment on a Japanese robot satellite ETS-VII," *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, Vol. 2, 2000, pp. 1169–1174 vol.2.

[16]Oda, M., "Space robot experiments on NASDA's ETS-VII satellite-preliminary overview of the experiment results," *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, Vol. 2, IEEE, 1999, pp. 1390–1395.

[17]Whelan, D. A., Adler, E. A., Wilson, S. B., and Roesler, G. M., "DARPA Orbital Express program: effecting a revolution in space-based systems," *Small Payloads in Space*, Vol. 4136, International Society for Optics and Photonics, 2000, pp. 48–57.

[18]Friend, R. B., "Orbital express program summary and mission overview," *Sensors and Systems for space applications II*, Vol. 6958, International Society for Optics and Photonics, 2008, p. 695803.

[19]Ogilvie, A., Allport, J., Hannah, M., and Lymer, J., "Autonomous satellite servicing using the orbital express demonstration manipulator system," *Proc. of the 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS'08)*, 2008, pp. 25–29.

[20]Coleshill, E., Oshinowo, L., Rembala, R., Bina, B., Rey, D., and Sindelar, S., "Dextre: Improving maintenance operations on the International Space Station," *Acta Astronautica*, Vol. 64, No. 9, 2009, pp. 869 – 874.

[21]Hirzinger, G., Landzettel, K., Reintsema, D., Preusche, C., Albu-Schäffer, A., Rebele, B., and Turk, M., "Rokviss - robotics component verification on ISS," *Proceedings of 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, 2005.

[22]"Robotic Refueling Mission 3," https://sspd.gsfc.nasa.gov/RRM3.html.

[23]Seedhouse, E., *SpaceX: making commercial spaceflight a reality*, Springer Science & Business Media, 2013.

[24]Kessler, D. J., "Collision Cascading: The Limits of Population Growth in Low Earth Orbit," *Adv. Space Res.*, Vol. 11, No. 12, 1991.

[25]Kessler, D. J. and Cour-Palais, B. G., "Collision Frequency of Artifical Satellites: The Creation of a Debris Belt," *Journal of Geophysical Research*, Vol. 83, No. A6, 1978.

[26]Artigas, J., De Stefano, M., Rackl, W., Lampariello, R., Brunner, B., Bertleff, W., Burger, R., Porges, O., Giordano, A., Borst, C., et al., "The OOS-SIM: An on-ground simulation facility for on-orbit servicing robotic operations," *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 2854–2860.

[27]Weber, D., Falcone, R., Gnat, M., Hauke, A., and Huber, F., "Technical studies for operations with real-time communications in robotic missions," *SpaceOps 2016 Conference*, 2016.

[28]Hubert, B., "Linux Advanced Routing and Traffic Control HOWTO," http://lartc.org/howto/, 2012.

[29]Hauke, A. and Barkasz, E., "Multi-Mission Support with WARP," 06 2012.

[30]"TM SPACE DATA LINK PROTOCOL," *BLUE BOOK - CCSDS 132.0-B-2*, 2015.

[31]"TC SPACE DATA LINK PROTOCOL," *BLUE BOOK - CCSDS 232.0-B-3*, 2015.

[32]"TC SYNCHRONIZATION AND CHANNEL CODING," *BLUE BOOK - CCSDS 231.0-B-2*, 2010.

[33]"TM SYNCHRONIZATION AND CHANNEL CODING," *BLUE BOOK - CCSDS 131.0-B-3*, 2017.

[34]"SPACE PACKET PROTOCOL," *BLUE BOOK - CCSDS 133.0-B-1*, 2012.

[35]"Space Link Extension Services - Executive Summary," *GREEN BOOK - CCSDS 910.0-G-2*, 2016.